A Multi-tiered Model for Context-Aware Systems

Abstract
Context awareness is considered one of the most important challenges to be tackled in the field of ubiquitous computing (ubicomp). In this perspective, this paper describes a general model for context-aware systems. The model is organized in multiple tiers, each one addressing a specific design characteristic related to the area. The proposition can be applied in the design and assessment of context-aware systems.

Author Keywords
Context awareness; sensors; context management.

ACM Classification Keywords
D.2.11. Software Architectures: Domain-specific architectures.

Introduction
Context awareness is one of the challenges that must be addressed when developing ubiquitous computing (ubicomp) systems. The idea consists in the perception of characteristics related to the users and their surroundings. These characteristics are normally referred to as context, i.e. any information that can be used to describe the circumstances concerning an entity (persons, places, or objects) [1].
Context is generally acquired using embedded computers or sensors. However, most devices today cannot sense their environment, and neither can software react to these changes. To help dealing with these and other limitations related to the subject, there are special kinds of software commonly known as context-aware systems [2]. They are in charge of inferring context to supply information when the availability of services is limited or intermittent.

Based on perceived context, the software can modify the behavior of the system. This process, in which software modifies itself according to sensed data, is named adaptation. This constitutes the core of a ubicomp challenge identified as context management. This article does not include an analysis of the adaptation process, since many other articles cover this subject [1,3].

In this article, we present a general model that can be used to assess and design context-aware systems. These systems usually have a set of services to deal with context, including a service that can notify a component in the occurrence of some event, a mechanism to find suitable information or services, the conversion of low-level data into high-level information, and the aggregation of context information to generate a more precise or detailed context.

**The Multi-tiered Model**

The proposed model is presented in Figure 1, which includes the main components found in context-aware systems. Various systems have been proposed, which differ in architecture, scope, aim, and also in the name they use for the tiers. Additionally, some systems do not include all the layers presented in our model, while others integrate some tiers into fewer components.

The bottom-most tier (sensor) consists of a collection of sensors that gather the raw data. These sensors are coordinated, parameterized, and controlled by the acquisition layer. The next tier carries out the transformation of data obtained from sensors into higher-level information. The following layer is responsible for synthesis, i.e. aggregating the context information, generating a more detailed context. Then, the model presents a discovery layer that is employed in the finding of sensors and other resources. The sixth tier represents a storage that accumulates the context data. Normally, there is inference atop of this data (or at least straightforward queries) which is accomplished by the subsequent tier, named query / inference. Next, the subscription and delivery layer offers event-communication mechanisms for notification of context events. And finally, the top-most tier represents the context-aware application.

The model shows two common flows of interaction among the tiers. The first one, represented by a line with a filled shape on both ends, represents the context subscription action and the subsequent occurrence of the event. Note that some tiers are not used (represented by a dotted line) in the top-down path, only in the return of the gathered context data.

Furthermore, there are tiers that are not used in this flow. The second showed flow, denoted by a line with a hollow shape on both ends, indicates a query on the database. This is usually used to obtain historical data or, in some systems, to apply a reasoning engine to the available context information.
It is a noteworthy matter that these two flows are not
the only possibilities in a context-aware system;
however, they represent the two most common
operations related to the subject. The next subsections
describe the main design principles involved in each
tier.

Sensors
There are three types of sensors that can be used in a
context-aware system: physical (or hardware) sensors,
virtual (or software) sensors, and logical sensors,
indicating that data can be provided either by a
physical or a virtual sensor. Individual nodes, usually
physical sensors, can be combined in a more complex
arrangement, generating a sensor network, which
mingles relatively simple sensors with real-time, low-
level manipulation and analysis.

Acquisition
The main purpose of this tier is to isolate the top layers
of the system from the complexity of gathering data.
Some systems do not present acquisition as a separate
layer, reducing the possibility of reusing sensors, and
jointly handling both the obtaining of data and its use
or representation. In the design of the acquisition tier
we must deal with the following concerns related to
sensors: installation, configuration, ways of
communication, and type of sensed data.

Transformation
This tier, also called context interpretation, transforms
the information received from the acquisition layer into
a machine processable format. The main concern in this
process is which context model to use in the
representation of context.

There are various data-structure modeling approaches
employed in the representation of context [4]: key-
value models – using pairs of values to represent
context; markup-scheme models – applying a
hierarchical data structure with markup tags, such as
XML; graphical models – utilizing a visual
representation, e.g. UML; object-oriented models –
exploiting OOP techniques in context representation;
logic-based models – employing a logical definition,
defining context as facts, expressions, and rules;
ontology-based models – applying ontology for
denoting context.

Synthesis
Synthesis or aggregation is the process of composing
individual context information related to a specific
entity. The module identifies different sources of
information and combines contexts to produce a result
that is more precise and easier to use. The main design
principle in this tier is related to the process of
aggregation: what the rules are, or what the domain is,
for combining context information.

Discovery
This tier is responsible for dynamic search and finding
resources at run time. When addressing context
awareness, the resources we are particularly interested
in are sensors. Some systems offer services or
components that accomplish this process, while others
need to employ built-in sensors or to rely on a pre-
configuration. In this case, failures and the addition of
new sensors need to be manually managed. One
characteristic of the discovery mechanism is the
method used for detecting the dynamic availability of a
resource. It is common to employ leases or some
pooling mechanism. Another design principle is the method that is used to look up resources.

**Storage**
The storage tier keeps the context information. Some systems maintain historical context data that “may be used to establish trends and predict future context values”. The main design principle is related to distribution and placement. The solutions vary from centralized to totally distributed storage. Trade-off alternatives are also possible, such as employing hierarchical or partially distributed solutions.

**Query and inference**
This layer comprises the management of context information. It can vary from simple mechanisms for querying the data, to powerful reasoning, including the inference of deduced context. This option constitutes the first design characteristic of the tier: presence or absence of an inference engine and a knowledge base. This characteristic has an influence on the type of generated context. The tier could infer only explicit context, or it could also infer implicit context, i.e. context generated from reasoning.

**Subscription and delivery**
Normally associated with context-aware systems, is the capability of subscribing to specific context change. Typically, this operation is based on a publish-subscriber model. Because of that, we chose subscription and delivery for the tier’s name. This layer gives asynchronous communication to the context-aware system, while the former tier generally constitutes a synchronous operation.

When subscribing to a context change, some systems allow the specification of certain conditions to be observed for the occurrence of an event. Furthermore, particular attributes that are of interest may be indicated. Another design issue is whether it is possible to subscribe to changes related to specific entities or only to individual sensors. This defines the subscription element: a sensor or an entity-related context.

**Application**
The last tier is represented by the context-aware application. The main design principle in this layer is connected to the way programs make use of context. This is strictly related to the context management method, and particularly to the adaptation strategy.

**Conclusion**
We believe that the proposed model could be useful in assessing and designing context-aware systems. The model covers many aspects related to the field. To evaluate our model, we are currently developing a software infrastructure specifically aimed at context awareness.

**References**


